SPECIFICATION

Electronic Version 1.2.8

Stylesheet Version 1.0

DNA Intrusion Detection Method

Summary of Invention

[0001] Loscocco, Smalley, Muckelbauer, Taylor, Turner and Farrell (2000) stated that no single technical security solution could provide total system security; a proper balance of security mechanisms must be achieved. Each security mechanism provides a specific security function; and should be designed to only provide that function. It should rely on other mechanisms for support and for required security services. In a secure system, the entire set of mechanisms complement each other so that they collectively provide a complete security package. Systems that fail to achieve this balance will be vulnerable.

[0002] Current art of intrusion detection focuses on signature examination or cataloguing patterns of *self* behavior so that *non-self* activity can be detected. It is the goal of the present invention to provide a method of "creating self-objects to allow the system to identify non-self objects."

[0003] The DNA Intrusion Detection Method is organized into three general phases. The DNA Definition phase defines the DNA Pattern, the environment in which it belongs, the process for injecting it into the objects, and a storage facility designated as the External Data Storage Structure (EDSS). The DNA Creation phase injects the DNA Pattern into the computer system objects, creates a database of new objects and creates an information database. The DNA Authentication phase authorizes an object for execution by the computer system. The processes in the DNA Definition phase are executed once while the DNA Creation and DNA Authentication phases are executed continuously as new objects are encountered and existing objects are prepared for execution.

[0004]

Selected objects processed through the DNA Creation phase contain identifiers

that connect them to a unique DNA Scope Set. Execution of those objects is accomplished only through the DNA Authentication phase. While this method does not restrict forces from placing unauthorized objects in the system, it will trap those objects and allow the system administrators to review and analyze them prior to execution.

Detailed Description

[0005] *Background*

[0006] Denning (1999) stated that the use of standard protocols allows interoperability across networks. While this facilitates communication and sharing, it also has drawbacks. Vulnerabilities can be pervasive across computer platforms and organizations, allowing thousands of systems to be swept up in a single attack.

[0007] Loscocco et al. (2000) stated that the increased awareness of the need for security has resulted in increased of efforts to add security to computing environments. However, these efforts suffer from the flawed assumption that security can be provided adequately in an application space without certain security features in the operating system. In reality, operating system security mechanisms play a critical role in supporting security at higher levels.

[0008]

Amoroso (1999) described signatures of abnormal behavior to detect possible intrusions. Some intrusion detection indicators are repetition of a suspicious action; mistyped commands or responses during an automated sequence; exploitation of known vulnerabilities; directional inconsistencies in inbound or outbound packets; unexpected attributes of some service request or packet; unexplained problems in some service request, system, or environment; and suspicious character traffic on a network. Even when a computer system is equipped with stringent authentication procedures and firewalls it is still susceptible to hackers who take advantage of system flaws and social engineering tricks (Goan, 1999). Goan continued, stating that the most obvious conclusion that can be drawn from two decades of research is that there are no easy answers, no silver bullets. Effective intrusion detection capability remains elusive as computing environments become more complex and crackers continually adapt their techniques to overcome innovations in computer security.

09783803.061401

Additionally, network administrators have been slow to adopt intrusion detection technology due to, among other reasons, excessively high false alarm rates associated with existing tools. These false alarms require a high degree of human analysis, thus reducing existing intrusion detection systems to the status of simple evidence sources. Given this observation, further advances in automated intrusion detection will require the development of new means of exploiting available evidence.

[0009] Computer viruses, worms and other devices are able to penetrate computer systems by becoming part of an operating system, application or data. When executed, these unauthorized agents have the potential to damage the host system and, using the authority of the host system, penetrate other systems. Password sub-systems, firewall sub-systems, intrusion detection systems and encryption, which are used to protect computer systems, are external agents that are designed to encapsulate the operating system, applications and data protecting them from intrusion. Dr. Stephanie Forrest of the University of New Mexico compared the process of computer system defense to the process used by living organisms to defend against diseases, viruses and other foreign agents (Forrest, Hofmeyr & Somayaji, 1997). Her thesis was to develop a methodology for identifying the *self* to use intrusion detection to detect *non-self* agents. Dr. Forrest suggested procedures for identifying the *self* by observing patterns of behavior of the system. An alternative to this external view is an operating system that contains its own self-defense mechanism.

[0010] This method creates a self-identity organization enabling the operating system to identify foreign agents automatically. This method will allow insertion of identification data into an object to identify uniquely the object to the operating system. This identification data is defined as a *DNA Pattern,* which is a sequence of identifier fields. Embedding an operating system DNA pattern into an object will differentiate it from all other objects of the same function in other operating system locations. Creation of a self-object will allow the system to identify foreign (or non-self) objects that were copied to the system without going through the DNA insertion process. This would eliminate viruses and Trojan horses from being executed without prior authorization.

[0011] *Description of Invention*

[0012] This following terms will be used in this section:

[0013] DNA Domain: An environment where computer system objects reside.

[0014] DNA Domain Administrator: An individual or group responsible for authorizing new objects to enter the DNA Domain.

[0015] DNA Object: A computer system's executables and non-executables that reside in the DNA Domain.

[0016] DNA Pattern: A sequence of identifier fields that will serve to create a unique copy of the object and create an ownership token between the object and the operating system; a collection of DNA Object properties that differentiate one DNA Scope Set from another in the DNA Domain.

[0017] DNA Scope Set: The set of DNA Objects residing in the DNA Domain having the same DNA Pattern.

[0018] DNA Scope Set Administrator: An individual or group responsible for authorizing new objects to enter the DNA Scope Set. Objects could be moved to the DNA Scope without the DNA Scope Set Administrator's authorization, but the DNA Intrusion Detection process will trap the unauthorized object before it's execution by the central processing unit (CPU).

[0019] DNA Steganographic Object: A DNA Object containing a DNA Steganographic Zone.

[0020] DNA Steganographic Zone: An area containing the DNA Pattern that is inserted into a DNA Object and is similar in appearance to the DNA Object.

[0021] *DNA Definition Phase*

[0022] This phase establishes the infrastructure of the DNA Intrusion Detection Method and is accomplished by completing five processes:

[0023] 1. Define the DNA Domain and DNA Scope Set (Process D1)

[0024] 2. Establish the DNA Pattern (Process D2)

[0025] 3. Define a Data Hiding Pattern (Process D3)

[0026] 4. Define the Process for Injecting the DNA Steganographic Zone into the DNA Object (Process D4)

[0027] 5. Define the EDSS (Process D5)

[0028] 1. Define the DNA Domain and DNA Scope Set (Process D1)

[0029] A computer system's executables and non-executables are defined as DNA Objects. The DNA Scope Set is defined as the set of DNA Objects having the same DNA Pattern. The DNA Domain is defined as an environment where multiple DNA Scope Sets co-exist. The DNA Pattern, therefore, is defined as a collection of DNA Object properties that differentiate one DNA Scope Set from another in the DNA Domain.

[0030] Example #1:

[0031] A simple example is a single stand-alone personal computer with an operating system and three application systems. Each application system has an exclusive set of DNA Objects. In addition, the operating system has its own set of DNA Objects. The DNA Domain, therefore, is defined as the entire set of objects across all the application systems and the operating system. Four DNA Scope Sets are defined: three for the application systems and one for the operating system. Likewise, there are four DNA Patterns in a one-to-one correspondence with a DNA Scope Set. An intrusion detection system using this organizational structure would be in a position to identify any object that was not previously approved by an authorized system user.

[0032] Let each a_i, for i = 1 to k, be a DNA Object of application system A. Then the DNA Scope Set of objects for application system A = $\{a_1, a_2, a_3, \ldots, a_k\}$.

[0033] Let each b_i, for i = 1 to l, be a DNA Object of application system B. Then the DNA Scope Set of objects for application system B = $\{b_1, b_2, b_3, \ldots, b_l\}$.

[0034] Let each c_i, for i = 1 to m, be a DNA Object of application system C. Then the DNA Scope Set of objects for application system C = $\{c_1, c_2, c_3, \ldots, c_m\}$.

[0035] Let each o_i for i = 1 to n be a DNA Object of the operating system. Then the DNA

09870895

Scope Set of objects for the operating system $O = \{o_1, o_2, o_3, \ldots, o_n\}$.

[0036] The DNA Domain is the union of the four sets of objects: {A, B, C, O}

[0037] Example #2:

[0038] There may be a situation where the operating system contains a set of shared DNA Objects that can be used by all of the application systems. Using the definitions from example #1, let the shared set of operating system DNA Objects be$\{o_3, o_4, o_5\}$. Then in this example the DNA Domain would be the same as in example #1, but the DNA Scope Sets for A, B and C would be expanded to include the three operating system DNA Objects.

[0039] $A = \{a_1, a_2, a_3, \ldots . a_k, o_3, o_4, o_5\}$

[0040] $B = \{b_1, b_2, b_3, \ldots . b_l, o_3, o_4, o_5\}$

[0041] $C = \{c_1, c_2, c_3, \ldots, c_m, o_3, o_4, o_5\}$

[0042] This is an example where computer systems violate the biological metaphor.

[0043] Example #3:

[0044] In another example, the designer may want to limit objects requiring a DNA Pattern. This may be due to system constraints, such as execution time or an application having a low risk of infection. If, from example #2, objects b_3, b_4 and b_5 from application system B do not require a DNA Pattern then the DNA Scope Set for $B = \{b_1, b_2, b_6, \ldots . b_l, o_3, o_4, o_5\}$ while the DNA Scope Sets for A, C and O would not change.

[0045] 2. Establish the DNA Pattern (Process D2)

[0046] The objective of this process is to define a set of object properties that will create a unique identity for objects across the entire DNA Domain. That is, if multiple systems are to be defined in a DNA Domain, the DNA Pattern of each system must be unique establishing a one-to-one correspondence between an object and a system in a DNA Domain. Some property examples are the system URL, the application's time-date stamp, the operating system code name, an application system code name or a

hash code.

[0047] Let each d_i for i from 1 to k be a property of a DNA Object for a given DNA Scope Set. Then a unique DNA Pattern is defined as a subset of those properties when compared to DNA Patterns of other DNA Scope Sets in the DNA Domain. For example, if $\{d_1, d_2, \ldots, d_k\}$ represents a set of DNA Object properties then a DNA Pattern may be defined as the set $\{d_5, d_8, d_{12}\}$ such that their property values in combination create a unique code across the DNA Domain.

[0048] Implementation of the DNA Pattern could be a string concatenation of the individual property values: $d_5 + d_8 + d_{12}$. From example #1 in process D1, the DNA Domain consists of four DNA Scope Sets (systems) resident on a single computer. Assume the CPU serial number is CPU73 and the names of the four DNA Scope Sets (three application systems and one operating system) are AA22, AB36, AC07 and ON12. A DNA Pattern can be defined as a string concatenation of the CPU serial number and the DNA Scope Set name. That is the four DNA Patterns are:

[0049] DNA Pattern for DNA Scope Set A = CPU73AA22

[0050] DNA Pattern for DNA Scope Set B = CPU73AB36

[0051] DNA Pattern for DNA Scope Set C = CPU73AC07

[0052] DNA Pattern for DNA Scope Set O = CPU73ON12

[0053] 3. Define a Data Hiding Pattern (Process D3)

[0054] There are two objectives of this process (D3) and the next process (D4): (1) to hide the DNA Pattern in the DNA Object and (2) to achieve the first objective while minimizing system overhead. Hiding the DNA Pattern in the DNA Object will deter its identification by a malicious intruder. The amount of effort expended to hide the DNA Pattern, however, will have an impact on system overhead because the hiding process is reversed during the DNA Authentication phase.

[0055]

Process D3 defines the a steganographic procedure for hiding the DNA Pattern in a DNA Steganographic Zone, which is defined as a new area that will be inserted into a DNA Object. The DNA Steganographic Zone must be crafted to be similar in

appearance to the DNA Object. For example, if a DNA Object is an executable then the DNA Steganographic Zone would contain what would appear to be machine code instructions.

[0056] Let each d_i for i from 1 to k be a property of a DNA Object selected to be in the DNA Pattern.

[0057] Let D represent the DNA Pattern set: $D = \{d_1, d_2, d_3, \ldots, d_k\}$.

[0058] Let each w_i for i from 1 to l be a subset of a representative generic zone that is similar in appearance to the DNA Object.

[0059] Let W represent that generic zone set: $W = \{w_1, w_2, w_3, \ldots, w_l\}$. Then the DNA Steganographic Zone, designated as W', is the *union* of D and W, where the elements of the set D are dispersed across W.

[0060] That is: $W' = \{w_1, w_2, w_3, \ldots, w_l, d_1, d_2, d_3, \ldots, d_k\}$

[0061] For example, if the elements of D and W are strings then W' could be defined as a concatenation of those strings: $W' = w_1 + d_1 + w_2 + d_2 + w_3 + d_3 + \ldots.. + w_k + d_k + w_{k+1} + w_{k+2} + \ldots + w_l$ where $l > k$.

[0062] Note that the *minimum* size of W' will be D with W defined as a null set. Larger DNA Steganographic Zones could be defined based on the efficiency of the extraction process. If l is much larger than k then the DNA Pattern can be hidden among the innocuous data. That is, if the size of D is small relative to W, then D could be hidden by breaking it up into pieces and distributing it across W creating a form of a subliminal channel. Also note that, prior to creating W', W could be reorganized as new elements. This, however, would add another step to the creation/authentication phases.

[0063] Depending upon the nature of the system, the designer may want to encrypt the DNA Pattern prior to inserting it into the DNA Steganograpic Zone. Cryptography provides functions that encrypt and decrypt data. For example, let function f be an encryption function, let x represent the data, which is called cleartext, to be encrypted, and let x' be the results of the encryption function. Then $x' = f(x)$, which is known as ciphertext. Cryptography developed a function, denoted as g, that reverses

the effect of f on x such that: $x = g(f(x))$. The example from above with the elements of D encrypted would be displayed as: $W' = w_1 + f(d_1) + w_2 + f(d_2) + \ldots + w_k + f(d_k) + w_{k+1} + w_{k+2} + \ldots + w_l$ where $l > k$.

[0064] 4. Define the Process for Injecting the DNA Steganographic Zone into the DNA Object (Process D4)

[0065] Execution of this process inserts the DNA Steganographic Zone (designated as W') into the DNA Object (designated as P) to create the DNA Steganographic Object (designated as P'). This is another set operation where P' is the union of W' and P.

[0066] Let each p_i for i from 1 to p be a subset of the DNA Object.

[0067] Let P represent the DNA Object set: $P = \{p_1, p_2, p_3, \ldots., p_p\}$.

[0068] In addition, from above the DNA Steganographic Zone is defined as:

[0069] $W' = \{w_1, w_2, w_3, \ldots, w_l, d_1, d_2, d_3, \ldots, d_k\}$

[0070] Then P' is the DNA Steganographic Object set:

[0071] $P' = \{p_1, p_2, p_3, \ldots., p_p, w_1, w_2, w_3, \ldots, w_l, d_1, d_2, d_3, \ldots, d_k\}$

[0072] If these set elements are strings then P' could be defined as a concatenated string: $P' = p_1 + p_2 + p_3 + \ldots.. + p_i + w_1 + d_1 + w_2 + d_2 + w_3 + d_3 + \ldots.. + w_k + d_k + w_{k+1} + w_{k+2} + \ldots + w_l + p_{i+1} + p_{i+2} + p_{i+3} + \ldots. + p_p$

[0073] The combined effect of processes D3 and D4 is to define a new DNA Steganographic Object (P') that is the combination of the original DNA Object (P), the generic zone set (W) and the DNA Pattern (D). In summary, P' is the union of the sets P, W and D. If implemented as character strings then the DNA Pattern, which may be encrypted, is split into pieces and inserted into a DNA Steganographic Zone, and the DNA Steganographic Zone is split into pieces and inserted into the DNA Object creating the DNA Steganographic Object.

[0074] 5. Define the EDSS (Process D5)

[0075] The primary purpose of the EDSS is to maintain the list of DNA Steganographic Objects and, at minimum, this structure would contain the DNA Steganographic

Object names and the DNA Pattern. The EDSS may also be used to store key information related to the definitions created during the DNA Creation phase. This information will be used in the DNA Authentication phase to extract the DNA Pattern and DNA Object from the DNA Steganographic Object. Key items stored would be the DNA Pattern, or the decryption key of the DNA Pattern if encrypted; the location and embedding configuration of the DNA Steganographic Zone in the DNA Steganographic Object; and the location and embedding configuration of the DNA Pattern in the DNA Steganographic Object.

[0076] For example, if P' is defined as the following character string: $P' = p_1 + p_2 + p_3 + \ldots\ldots + p_i + w_1 + d_1 + w_2 + d_2 + w_3 + d_3 + \ldots\ldots + w_k + d_k + w_{k+1} + w_{k+2} + \ldots + w_l + p_{i+1} + p_{i+2} + p_{i+3} + \ldots. + p_p$ the system would need to know the starting address of the Steganographic Zone (w_1) and its length in order to extract it from the DNA Steganographic Object. Also, each w_i must contain the address and length of the next one (w_{i+1}).

[0077] *DNA Creation Phase*

[0078] Once the environment has been established, through the DNA Definition phase, the system and its administrator can now use the DNA Creation phase to inject the appropriate DNA Pattern into selected objects of the DNA Domain. The DNA Creation phase injects DNA Objects from the DNA Domain with the DNA Pattern and creates the DNA Scope Set consisting of DNA Steganographic Objects. The six tasks of this phase are summarized below:

[0079] 1. Select a DNA Object from the DNA Domain

[0080] 2. Retrieve the DNA Pattern from the EDSS

[0081] 3. Insert the DNA Pattern into a DNA Steganographic Zone based on the data hiding pattern specified in D3

[0082] 4. Insert the DNA Steganographic Zone into the DNA Object based on the data hiding pattern specified in D4

[0083] 5. Store the DNA Steganographic Object in the system resource library, and move the original DNA Object off-line

[0084] 6. Store the DNA Object's control information in the EDSS

[0085] 1. Select a DNA Object from the DNA Domain

[0086] This methodology enables a two-stage authorization of new objects. An object must first enter the DNA Domain where the DNA Domain Administrator approves its movement into the DNA Domain becoming a DNA Object. The second stage authorization is performed by the DNA Scope Set Administrator who authorizes the object to be part of the DNA Scope Set. Input to this phase is a DNA Object and output is a DNA Steganographic Object.

[0087] 2. Retrieve the DNA Pattern from the EDSS

[0088] The DNA Creation phase identified certain properties of the each DNA Object to consist of the DNA Pattern. This task retrieves the DNA Pattern from the EDSS. Another security level may be designed into this process by encrypting the DNA Pattern before storing it on the EDSS, which would have been defined during the DNA Creation phase.

[0089] 3. Insert the DNA Pattern into a DNA Steganographic Zone based on the data hiding pattern specified in D3

[0090] Process D3 of the DNA Design phase may require the DNA Pattern (or its encrypted version) to be split into subsets. In addition, the same process defined how to embed the DNA Pattern into a generic steganographic zone creating the DNA Steganographic Zone. This task performs that function.

[0091] 4. Insert the DNA Steganographic Zone into the DNA Object based on the data hiding pattern specified in D4

[0092] Process D4 of the DNA Design phase may also require that the DNA Steganographic Zone be split into subsets. In addition, the same process defined how to embed the DNA Steganographic Zone into the DNA Object creating the DNA Steganographic Object. This task performs that function.

[0093] 5. Store the DNA Steganographic Object in the system resource library, and move the original DNA Object off-line

10409438.060401

[0094] A new form of the DNA Object, the DNA Steganographic Object, is now unique across the DNA Domain. The DNA Steganographic Object is moved to the system resource library. The unprotected version of the executable, the DNA Object, should be moved off-line to a secured location so that an unauthorized process cannot execute it on this CPU.

[0095] 6. Store the DNA Object's control information on the EDSS

[0096] The DNA Authentication phase will need several essential elements to extract the DNA Pattern from the DNA Steganographic Zone as well as recreating the DNA Object. Those elements are stored in the EDSS.

[0097] *DNA Authentication Phase*

[0098] This phase extracts the DNA Pattern from the DNA Steganographic Object and compares it to what the DNA Pattern should be, which is stored on the EDSS. Matched (authenticated) objects are forwarded to the operating system for processing. Notification is forwarded to the system"s administrator when unmatched objects are requested for execution. This process occurs in real-time and takes place just prior to the object"s execution. An object that successfully completes this phase will be sent to other operating system components for execution. The four tasks are summarized below:

[0099] 1. Locate the EDSS record and DNA Steganographic Object

[0100] 2. Extract the DNA Steganographic Zone from the DNA Steganographic Object recreating the DNA Object

[0101] 3. Extract the DNA Pattern, which may be encrypted, from the DNA Steganographic Zone

[0102] 4. Compare the extracted DNA Pattern to the system's definition of the DNA Pattern

[0103] 1. Locate the EDSS record and DNA Steganographic Object

[0104] Given a request to execute an object through the normal operating system and application security processes, the EDSS record corresponding to the object"s name is

retrieved. It contains key information for extracting the DNA Steganographic Zone from the DNA Steganographic Object and the DNA Pattern from the DNA Steganographic Zone. If the object is not found on the EDSS or the DNA Steganographic Object is not found, then this request is suspect and the DNA Scope Set Administrator is notified. This may be a new authorized object, which will need to be sent to the DNA Creation phase for processing. If not, the DNA Scope Set Administrator has the option to reject the object or begin an analysis to find out more information about the object.

[0105] 2. Extract the DNA Steganographic Zone from the DNA Steganographic Object recreating the DNA Object

[0106] Given the control information on the corresponding EDSS record, specifically the starting location of the DNA Steganographic Zone and its disbursement pattern across the DNA Steganographic Object, the DNA Steganographic Zone is extracted from the DNA Steganographic Object. This process also recreates the DNA Object.

[0107] 3. Extract the DNA Pattern, which may be encrypted, from the DNA Steganographic Zone

[0108] The EDSS record also contains the disbursement pattern and, if encrypted, the decryption key of the DNA Pattern in the DNA Steganographic Zone.

[0109] 4. Compare the extracted DNA Pattern to the system's definition of the DNA Pattern

[0110] The system is ready to compare the DNA Pattern extracted from the DNA Steganographic Object with the original DNA Pattern. For DNA Patterns that match, the system recognizes the object as *self* and authorizes the DNA Object for execution. For the DNA Patterns that do not match, the object is treated as a *non-self* object and the DNA Scope Set Administrator is notified. The owner may want to add this new object to the DNA Scope Set or begin an analysis procedure to find out more information about the object.

[0111] *References*

[0112] Amoroso, E. (1999). *Intrusion Detection: An Introduction to Internet Surveillance,*

Correlation, Trace Back, Traps, and Response. Sparta, NJ: Intrusion.Net Books.

[0113] Anonymous. (1998). *Maximum Security: A Hacker"s Guide to Protecting Your Internet Site and Network: Second Edition.* Indianapolis, IN: Sams Publishing.

[0114] Denning, D. E. (1999). *Information Warfare and Security.* Reading, MA: Addison-Wesley.

[0115] Forrest, S., Hofmeyr, S., & Somayaji, A. (1997). Computer immunology. *Communications of the ACM*, 40 (10), 88-96.

[0116] Goan, T. (1999, July). A cop on the beat: Collecting and appraising intrusion evidence. *Communications of the ACM*, 42 (7), 46-52.

[0117] Kahn, D. (1996). *The Codebreakers: The Story of Secret Writing*. New York, NY: Scribner.

[0118] Loscocco, P. A., Smalley, S. D., Muckelbauer, P. A., Taylor, R. C., Turner, S. J. &, Farrell, J. F. (2000, October 17). The inevitability of failure: The flawed assumption of security in modern computing environments. *National Security Agency*. http://www.esecurityonline.com/library/whitepapers2.asp

[0119] Petitcolas, F. A. P. (1999) Introduction to information hiding. In S. Katzenbeisser & F. A. P. Petitcolas (Eds.), *Information Hiding Techniques for Steganography and Digital Watermarking*, 1-14. Boston, MA: Artech House.

[0120] Schneier, B. (1996). *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. (2nd ed.). New York, NY: John Wiley & Sons.

10470907.060501